Mugurași

Autor: stud. Ignat Alex-Matei, Universitatea Babeș-Bolyai, Cluj-Napoca

Numărul de operații poate să fie destul de mare, iar atribuirea directă poate dubla lungimea unui șir, deci lungimea șirului poate să crească exponențial în funcție de N. Totodată, lungimile șirurilor din atribuiri și a șirului pe care trebuie să-l căutăm sunt cel mult 10, deci aceste lungimi nu vor influența foarte mult complexitatea finală. Atribuirea directă este simplă de calculat indiferent de restricții, problema adevărată fiind optimizarea atribuirii din memorie. Cu câteva observații cheie și strategii simple, putem să determinăm rezultatul cerut într-un mod eficient.

În continuare, vom aborda strategiile necesare fiecărui subtask.

Subtask 1. $|X| = 1$ și $|S| = 1$.

Pentru acest subtask, suntem interesați de frecvența unei singure litere S, iar numărul de variabile care pot să apară în operații este cel mult 26. Putem reține o frecvență pentru fiecare variabilă într-un vector. Pentru o atribuire directă, actualizăm frecvența cu numărul de caractere al șirului atribuit. Pentru o atribuire din memorie, actualizăm frecvența cu suma frecvențelor din cele două variabile.

Complexitate: $O(N)$

Subtask 2. $|S| = 1$.

Suntem interesați în continuare de frecvența unei singure litere S, însă din cauza lungimii denumirii variabilelor nu mai putem folosi un simplu vector. Pentru a reține eficient frecvența unei variabile, putem folosi un *map* (sau un *unordered_map* pentru operații în timp constant) din *STL* în care cheile ne sunt denumirile variabilelor reținute prin *string-uri*, iar valoarea asociată este frecvența literei. Actualizările frecvențelor se fac prin aceeași logică prezentată la *Subtask 1.*

Complexitate: $O(N)$ sau $O(N \cdot log\, N)$

Subtask 3. $N \leq 15$.

Din moment ce numărul de operații este mic, putem reține șirurile de caractere pentru fiecare variabilă în parte, lungimea unui șir la finalul operațiilor fiind cel mult $10 \cdot 2^{14}$, însă în practică nu toate variabilele pot să aibă această lungime. La finalul operațiilor, facem o singură parcurgere prin șirul din variabila V și determinăm numărul de apariții al lui S.

Complexitate: $O(2^N \cdot |S|)$

Subtask 4. $|X| = 1$.

Lungimea denumirii unei variabile fiind 1, avem cel mult 26 de variabile. Însă, nu putem reține întreg șirul de caractere, acesta fiind prea mare. Facem următoarea observație care pe baza căreia vom determina ușor rezultatul:

Observație. Numărul de apariții al unui șir în concatenarea a două șiruri este egal cu suma numărului de apariții al cuvântului în primul șir, numărului de apariții al cuvântului în al doilea șir și numărului de apariții al cuvântului având o parte în primul șir și o parte în al doilea șir.

Pe scurt, fie *nrap(W, S)* o funcție care determină numărul de apariții al șirului S în șirul W. Dacă avem concatenarea șirurilor A și B și ne interesează numărul de apariții al unui șir S, atunci numărul de apariții va fi egal cu:

$$nrap(A*B,\ S) = nrap(A,\ S) + nrap(B,\ S) + nrap(sufix(A)*prefix(B),\ S),$$

unde operatorul $*$ reprezintă operația de concatenare a două șiruri.

Presupunând că știm numărul de apariții al lui S în șirul A și în șirul B, rămâne de rezolvat o singură problemă: care ar fi sufixul lui A și prefixul lui B care ne interesează?

Singurele apariții pe care vrem să le determinăm sunt cele pe care nu le avem integral în A sau integral în B, deci o parte din șir aparține sufixului lui A și o parte din șir aparține prefixului lui B. Deci, ne interesează doar ultimele $|S| - 1$ litere din șirul A și primele $|S| - 1$ litere din șirul lui B. Determinarea numărului de apariții în plus pe lângă cel din A și din B depinde așadar doar de lungimea șirului S.

Din moment ce numărul de variabile este mic, putem reține într-un vector (sau în 3 vectori separați) pentru fiecare variabilă numărul de apariții al șirului cerut, sufixul și prefixul de cel mult $|S| - 1$ caractere. La o atribuire din memorie asupra unei variabile x și concatenarea a două variabile a și b, obținem următoarea relație:

$$prefix_x = prefix_a$$
(dacă nu am ajuns la $|S| - 1$ caractere, mai adăugăm și din $prefix_b$)

$$sufix_x = sufix_b$$
(dacă nu am ajuns la $|S| - 1$ caractere, mai adăugăm și din $sufix_a$)

$$aparitii_x = aparitii_a + aparitii_b + aparitii(sufix_a + prefix_b)$$

Singura parte din actualizare care trebuie calculată este numărul nou de apariții. La final, putem accesa direct rezultatul pentru variabila V.

Complexitate: $O(N \cdot |S|^2)$

Subtask 5. Fără restricții suplimentare.

Combinând ideile de la *Subtask 2* și *4*, putem folosi un *map* sau un *unordered_map* din *STL* în care să ne asociem la fiecare variabilă reținută ca *string* o structură ce conține toate cele trei informații necesare: numărul de apariții al lui S, sufixul și prefixul.

Complexitate: $O(N \cdot |S|^2)$ sau $O(N \cdot log\, N \cdot |S|^2)$

Notă.

Numărul de apariții poate să fie foarte mare, așa că trebuie să folosim variabile declarate pe 64 de biți (*long long*) pentru reținerea acestuia.

Deși operațiile cu *string-uri* sunt liniare în lungimea acestora, în practică se comportă mult mai bine, mai ales pentru *string-uri* de lungimi reduse.

Muguraș vă mulțumește pentru ajutor și vă recomandă din nou să nu învățați limbajul **Mac**.